AFRL-RI-RS-TR-2013-137

MODELING LARGE SCALE CIRCUITS USING MASSIVELY PARALLEL DESCRETE-EVENT SIMULATION

RENSSELAER POLYTECH INSTITUTE

JUNE 2013

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2013-137   HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ S /

RYAN LULEY
Work Unit Manager

/ S /

MARK LINDERMAN
Technical Advisor, Computing &
Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)* JUNE 2013	**2. REPORT TYPE** FINAL TECHNICAL REPORT	**3. DATES COVERED** *(From - To)* SEP 2011 – DEC 2012

4. TITLE AND SUBTITLE MODELING LARGE SCALE CIRCUITS USING MASSIVELY PARALLEL DISCRETE-EVENT SIMULATION	**5a. CONTRACT NUMBER** FA8750-11-2-0065
	5b. GRANT NUMBER N/A
	5c. PROGRAM ELEMENT NUMBER 63788F
6. AUTHOR(S) Christopher Carothers and Elsa Gonsiorowski	**5d. PROJECT NUMBER** T3PD
	5e. TASK NUMBER RP
	5f. WORK UNIT NUMBER IC
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Rensselaer Polytechnic Institute 110 8th Street Troy, NY 12180	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RITB 525 Brooks Road Rome NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RI
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-RI-RS-TR-2013-137

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# 88ABW-2013-2546
Date Cleared:29 MAY 2013

13. SUPPLEMENTARY NOTES

14. ABSTRACT

As computing systems grow to exascale levels of performance, the smallest elements of a single processor can greatly affect the entire computer system (e.g. its power consumption). As future generations of processors are developed, simulation at the gate level is necessary to ensure that the necessary target performance benchmarks are met prior to fabrication. The most common simulation tools available today utilize either a single node or small clusters and as such create a bottleneck in the development process. This paper focuses on the massively parallel simulation of logic gate circuit models using supercomputer systems. The focus of this performance study leverages the OpenSPARC T2 processor design and the PHDOLD benchmark model using Rensselaer's Optimistic Simulation System (ROSS). We conduct simulations of the crossbar component on a 24-core SMP machine, an IBM Blue Gene/L, and an IBM Blue Gene/Q. Using a single SMP core as the baseline, our performance experiments on 1024 cores of the Blue Gene/L demonstrate more than 131-times faster execution of the OpenSPARC model. We also present the performance results of ROSS executing the Time Warp synchronization protocol using up to 7.8M MPI tasks on 1,966,080 cores of the Sequoia Blue Gene/Q supercomputer system. For the PHOLD benchmark model, we demonstrate the ability to process 33 trillion events in 65 seconds yielding a peak event-rate in excess of 504 billion events/second using 120 racks of Sequoia.

15. SUBJECT TERMS
Circuit simulation , discrete event simulation , integrated circuit modeling , logic design , logic gates , logic simulation , microprocessor chips , parallel machines

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** **RYAN LULEY**
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	28	**19b. TELEPONE NUMBER (*Include area code)*** **N/A**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 SUMMARY

As computing systems grow to exascale levels of performance, the smallest elements of a single processor can greatly affect the entire computer system (e.g. its power consumption). As future generations of processors are developed, simulation at the gate level is necessary to ensure that the necessary target performance benchmarks are met prior to fabrication. The most common simulation tools available today utilize either a single node or small clusters and as such create a bottleneck in the development process. This paper focuses on the massively parallel simulation of logic gate circuit models using supercomputer systems. The focus of this performance study leverages the OpenSPARC T2 processor design and the PHOLD benchmark model using Rensselaer's Optimistic Simulation System (ROSS). We conduct simulations of the crossbar component on a 24-core SMP machine, an IBM Blue Gene/L, and an IBM Blue Gene/Q. Using a single SMP core as the baseline, our performance experiments on 1024 cores of the Blue Gene/L demonstrate more than 131-times faster execution of the OpenSPARC model. We also present the performance results of ROSS executing the Time Warp synchronization protocol using up to 7.8M MPI tasks on 1,966,080 cores of the Sequoia Blue Gene/Q supercomputer system. For the PHOLD benchmark model, we demonstrate the ability to process 33 trillion events in 65 seconds yielding a peak event-rate in excess of 504 billion events/second using 120 racks of Sequoia. This is by far the highest event-rate reported by any massively parallel simulation to date running the PHOLD benchmark. In terms of overall speedup, we report 97x performance improvement when scaling from 32,768 to 1,966,080 cores. This super-linear performance is attributed to significant cache performance improvements when running at peak scale. From these performance results, we devise a new, long range performance metric, called Warp Speed which grows linearly with an exponential increase in the PHOLD event-rate. At present, we are now at Warp Speed 2.7. It will be nearly 150 years before we expect to reach Warp Speed 10.0.

2.0 INTRODUCTION

As supercomputer systems approach exascale, the core count will exceed 1024 and number of transistors used in these designs will number in the 10's of billions. However, at the same time there is an increased need to understand the performance dynamics of these systems at the lowest levels in order to provide highly accurate performance and power utilization measures. For example, if a 25 MWatt supercomputer's power estimation is off by 5% that equates to 1.25 MWatt or $1.25 million per year of operation. Because of the size and operating costs of these systems, accurate estimates are needed.

Digital logic simulation is a well-known tool which can be used to address this problem. However, to date, gate-level simulation has been too computationally intensive to be an effective solution. As a first step, this paper focuses on the parallel simulation of the OpenSPARC T2 crossbar (CCX) component making use of current supercomputer systems. Using ROSS, we are able to investigate both conservative and optimistic parallel discrete-event simulation (PDES).

We include both strong scaling and weak scaling experiments, along with attempts to increase the efficiency of optimistic simulation. For the weak scaling experiments, a single crossbar is replicated many times and nominally connected across a few data lines. These experiments are conducted on two machines:

- A 24-core symmetric multiprocessing (SMP) machine with 2.667 GHz cores (baseline).
- An IBM Blue Gene/L with 700 MHz cores interconnected with fast communication networks.

We demonstrate the efficacy of using massively parallel systems to improve the performance from almost 900,000 events per second on a single SMP machine core to over 116,000,000 state-transition events per second on 1024 Blue Gene/L cores. This represents an improvement in performance of nearly 131 times. Additionally, we find that because of the low variance in timestamp increments within the gate model, conservative synchronization outperforms optimistic processing with reverse computation.

The key question is how much more scalability is possible? Within this report we address this question and also conduct a detailed experimental study of ROSS executing the PHOLD benchmark model on two systems: the two rack, 32,768 core Blue Gene/Q located at Rensselaer’s Computational Center for Nanotechnology Innovations (CCNI) supercomputer and Sequoia, a 120 rack, 1,966,080 core Blue Gene/Q supercomputer.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 The ROSS Framework

ROSS is a framework designed for parallel discrete-event simulation and is built upon Jefferson’s Time Warp. Each discrete object in the simulation is known as a Logical Process (LP). LPs communicate with each other through messages (also known as events). In this specific simulation, the LPs model gate objects. The messages sent between the LPs represent electrical signals between gates.

Unlike many other simulation systems, ROSS has no form of state-saving. While event reconstruction has been used with circuit simulation before, reverse computation is most effective for this model due to small message size. To revert the state of a gate, a reverse computation is used. Through reverse messages, the gate object must be able to undo any state changes which happened during forward event processing. Thus, for each forward event course of action, a corresponding reverse computation must be implemented.

The underpinnings of ROSS are written in ASNI C and designed for efficiency. This includes the reversible random number generator library based upon a Combined linear Congruential Generator. All memory is handled within ROSS itself, and is centered around managing the event objects. The overarching event list (for each processor) is a priority queue, implemented as either a Calendar Queue or a Splay-Tree. This research utilizes only the Splay-Tree implementation.

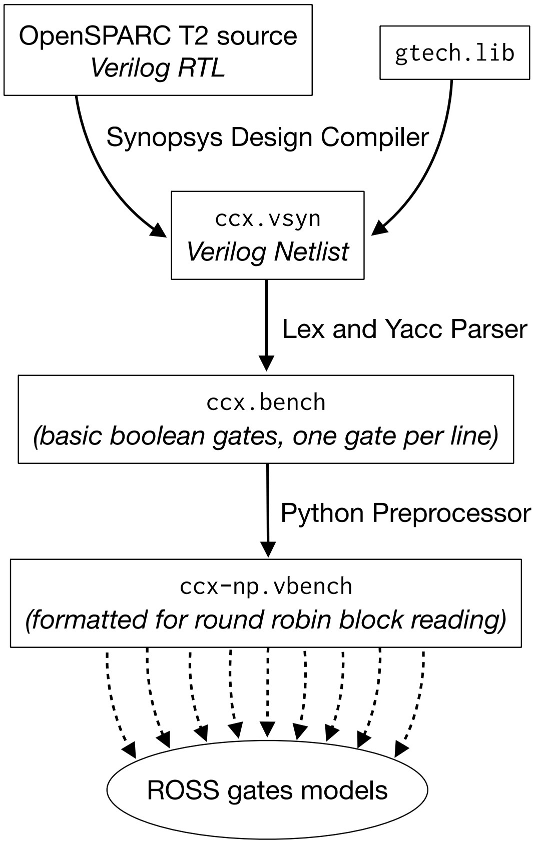

Figure 1 - Process for Transforming RTL Code to Machine Readable Gate Descriptions

3.2 The Data

The OpenSPARC T2 processor design is an excellent example of open-source hardware. It is a 64-bit, eight-core, microprocessor description in VHDL and Verilog. Using the Synopsys Design Compiler and scripts provided by the OpenSPARC code base, we were able to generate gate level descriptions for discrete sections of the processor. In this paper we focus solely on the CPU-cache crossbar, the processor component which links the eight processing units. This module contains over 200,000 gates.

The gate description is transformed and processed in many ways before it can be efficiently used in a simulation model. This process is described in Figure 1.

3.2.1 Source. The OpenSPARC T2 design is provided in Verilog Register Transfer Language (RTL). This flexible description is fairly high level, especially from a hardware perspective. It encodes modules at an abstract, logical level (Table 1).

Table 1 - Sample of Data Format for Source Description

```
module ccx (scan_in, scan_out, ...);
wire [1:] scan_in_buf;
input [1:] scan_in;
...
clkgen_ccx_cmp clk_ccx (...);
endmodule
```

3.2.2 Gate Level. Using the Synopsys Design Complier, the RTL source can be synthesized into a gate level description. At this level several high level modules can be synthesized into one flat netlist. This file format is still completely valid Verilog code. The module is defined with connection arguments and the netlist of its gates. For this research, we used the generic technology library (GTECH) (Table 2).

Table 2 - Sample of Data Format for Gate Level Description

```
module ccx (scan_in, scan_out, ...);
input [1:] scan_in;
...
GTECH_NOT U61 (.A(n1319), .Z(n94428));
GETCH_BUF U78 (.A(scan_out[1]), .Z(ccx_out[15]));
endmodule
```

3.2.3 Basic Boolean Gates. The GTECH standard cell library is provided by Synopsys. This gate library contains over 100 specialized gates, many of which operate at a higher level than the traditional Boolean logic level. To facilitate development, each GTECH module is further broken down into its basic gates. Of these basic gates, there are only 8 logical types: repeater (DFF), NOT, AND, NAND, OR, NOR, XOR, and XNOR. These gates are flexible and allow for a maximum of four inputs. Three additional types are available: input, output, and clock, and are kept as distinct types.

The transformation is accomplished by a Verilog parser provided by Li et al. This parser utilizes Lex and Yacc.

The resulting file format has one gate description per line. Input and output gates specify the purpose of a named connection. Each gate object is written as an assignment. The named output wire is the result of a gate function which takes named input wire arguments. At this level most of the original wire names are maintained, however, intermediate wires have been inserted by the parser (Table 3).

Table 3 - Sample of Data Format for Basic Boolean Gate Description

```
INPUT(scan_in[])
INPUT(scan_in[1])
OUPUT(scan_out[])
OUTPUT(scan_out[1])
...
n1319 = NOT(n94428)
ccx_rstg_out[15] = DFF(scan_out[1])
```

3.2.4 Machine Format. The final conversion step is simple processing to allow for efficient machine reading. At this point, almost all of the details of each gate are captured in a single line. However, gate names are still expressed as strings and must be converted to global identification numbers. For maximum compactness, line numbers correspond to the global id and the implicit order must be understood by the ROSS model.

The meaning of each number on a line in Table 4 is as follows:

- 0th number: Line number implies global ID
- 1st number: Output count
- 2nd number: Gate type
- 3rd – 6th number: Global IDs of any input connections

Table 4 - Sample of Data Format for Machine Readability

```
0 2
0 3 3
2 5 46762
1 7 126287 126288 126289 126290
1 6 126192 126274
12 4 29309
```

3.2.5 Mapping and Parallel I/O. When formatting data files for reading in parallel, one must ensure that individual blocks can be read efficiently by any single processor. Block reading means that each line must contain the same number of bytes, allowing for block size calculation at runtime. Thus, each line is filled with whitespace characters to ensure a consistent line length.

The final step for the data file is to ensure that the order matches the LP mapping at runtime. Due to the fact that the original crossbar component (in Verilog RTL form) had a high level module ordering, the final machine data file may have some overarching, implicit ordering. For small simulations on many processors (as in the strong scaling experiments) we attempt to balance the load across all processors which contain a part of a distinct crossbar. To create this balance, the gate objects are distributed, round robin style, among processors. Again, block reading by an individual processors means that the lines of the data file must also be rearranged in a round robin fashion.

Depending on the simulation, several processors may need to read a single data file. This is easily accomplished, in parallel, through the Message Passing Interface (MPI) standard. Within the `MPI_File_open` function, one can specify both the file mode and processor group. The MPI constants `MPI_MODE_RDONLY` and `MPI_COMM_SELF` indicate read-only mode on a single processor. This means, even when many processors are accessing the same file, each processor is given its own file handler. The final step of file reading involves the `MPI_File_read_at` function, with one function call for each line/gate to read. By using parallel reads, this method is both fast and efficient.

3.3 The Model

3.3.1 Gate Objects and Messages. The gate level simulation is executed at the LP level. Each LP models a single gate, sending messages to other gates (LPs). While some data is stored in the ROSS LP object (such as the global identification number of the LP), most information is stored in a state object. This struct defines each gate's type, its input and output connections, as well as the statistics for the individual gate. At a global level, gate functions are defined and are indexed by each unique gate type. These functions transform the input and output connection arrays and can be easily redefined for a different EDA gate library.

Within the ROSS framework the state object is transformed based upon received messages. Due to the use of reverse computation, there is only a current copy of each LP's state. The small messages represent (in this model) the electronic high/low or 1/0 signals between gates. Within ROSS, a message is processed by the forward event handler; if a rollback occurs, the same message is un-processed with the reverse event handler. This allows, by updating the message itself, for single values to be saved between forward and reverse computations.

The lifetime of a gate object within the ROSS framework is as follows:

1) Global Virtual Time < 0: Initialization via file input.
2) GVT = 0..10: Setup messages to link gates across the network.
3) GVT = 10..end: Simulation message processing with possibilities of rollbacks.
4) GVT > end: Wrap up and final statistic reporting.

3.3.2 Simulation Time. In the gate simulation, one unit of global simulation time represents one clock cycle. A basic assumption in this model is that each gate has a delay of one clock cycle. Within the clock cycle a single LP follows the basic timeline in Figure 2. All messages sent from one gate object to another are sent at time *x*.5. These messages have a staggered arrival time, and arrive within a 0.2 clock cycle window. The first message to arrive at a gate object triggers a self-update event, scheduled at the next half clock cycle. This self-scheduled event triggers the next round of inter-gate messages.

The lookahead value is the minimum value timestamp that an LP can use to schedule a future event. Within this model a lookahead value of 0.4 is used. Overall, the average timestamp increment is 0.5 clock cycles.

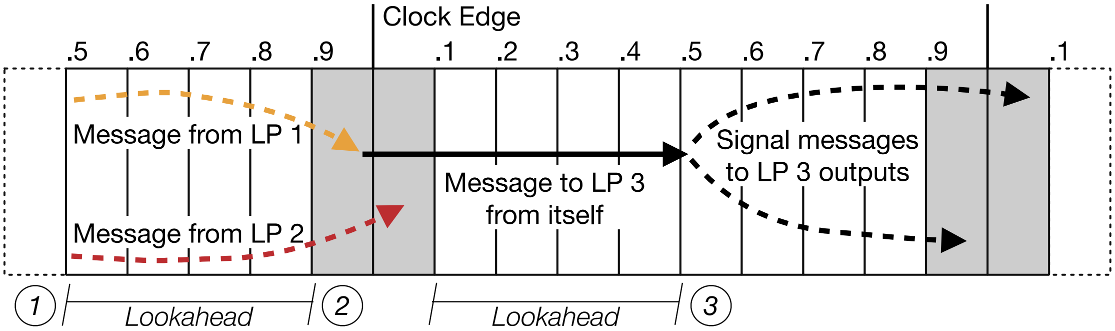

Figure 2 - Simulation Clock Cycle Time Slice

3.3.3 Large Circuits. The future of computing would appear to lie in parallel systems; however, parallelization is happening at the chip level. Single chips contain many computing cores. As the size and complexity of these chips grow, the size of the simulation models grows accordingly. Large scale simulations, on the order of one billion gates will be necessary to execute. The designs for these future technologies are not available to the community. Hence the only option for demonstrating our simulation on such a large scale circuit is to devise our own model for experimentation.

While advanced approaches to synthesis exist, the most basic and obvious method is duplication of an existing circuit. In order to efficiently create a large scale circuit for simulation, we duplicated the crossbar. In addition, some data lines (such as the CPU repeaters) are connected across crossbar instances. The overall duplication is grid based, with data line connections occurring along horizontal rows of crossbar instances, see Figure 3.

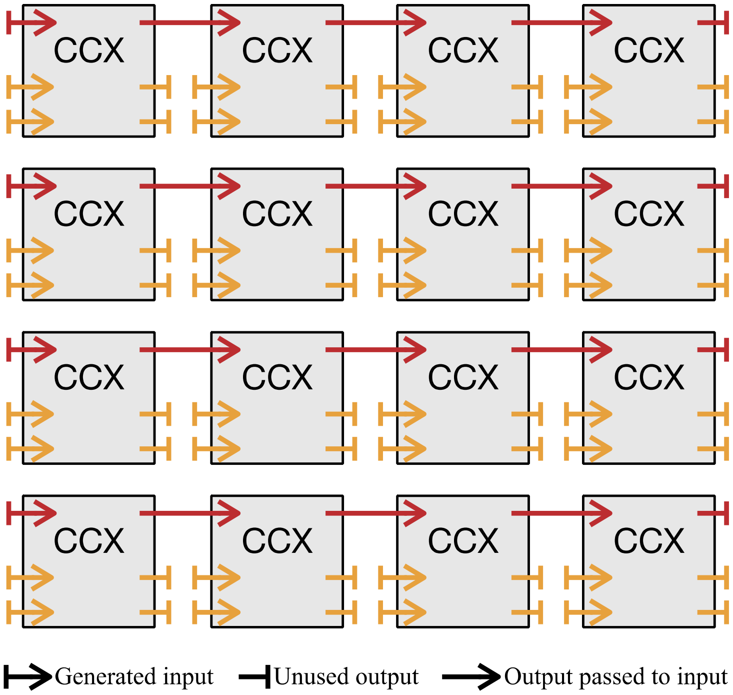

Figure 3 - A 4x4 Synthetic Circuit

3.4 The Blue Gene/Q Architecture

The Blue Gene/Q is the third generation system in the Blue Gene family with the Blue Gene/L and /P coming previously in 2004 and 2007, respectively. The Blue Gene/Q has made a substantial leap in computational capabilities over these previous generations. The largest systems are currently the 48 rack *Mira* system located at Argonne National Laboratory (ANL) and the 120 rack Sequoia system located at Lawrence Livermore National Laboratory (LLNL).

The Blue Gene/Q's A2 processor (one per Blue Gene/Q node) is a 45 nm chip compromised of 18 cores (PU's in the Figure) running at 1.6 GHz. The first 16 cores are exclusively used to execute user-level compute tasks while a 17th core is reserved to perform OS functionality. This design enables the use of a much more capable compute-node OS (e.g., supports mmap, shared libraries, etc.) without introducing OS jitter. Each core supports up to 4 hardware thread or MPI task contexts with each context having a full 32, 64-bit registers. Each core supports up to 4-wide double precision SIMD calculations. In terms of execution models, the Blue Gene/Q supports up to 64 MPI tasks per node (up to 4 tasks per core) or a mix of MPI tasks and pthreads. In total, the peak performance of the A2 processor is 204.8 gigaflops and consumes only 55 watts of power. A fully-connected cross bar switch connects all 16 cores to the 32 MB L2 cache (realized as 16, 2 MB L2 cache units), thus all processors have a uniform cache-memory access. The amount of memory is fixed on each compute node at 16 GB with a bandwidth of 42.6 GB/sec.

In contrast to the Blue Gene/P, which only has four PowerPC 450 cores operating an 850 MHz, resulting in 13.6 gigaflops, the Blue Gene/Q provides a 15x performance boost in peak FLOPS. However, a potential barrier to reaching that performance increase is the available node memory bandwidth. The Blue Gene/P provided 13.6 GB/sec which yields a memory bandwidth to FLOP ratio of 1. The Blue Gene/Q has a memory bandwidth to FLOP ratio of only 0.21. Lowering this ratio even further relative to that of the Blue Gene/P is the doubling of the address space size from 32 bits on the Blue Gene/P to 64 bits on the Blue Gene/Q. Thus, for pointer intensive applications like ROSS, the effective pointer access rate has only improved by a factor of 1.5x (((42.6 GB/s/64 bits)/(13.6 GB/s/32 bits)) in going from the Blue Gene/P to the Blue Gene/Q. To help mitigate the lower overall memory bandwidth to FLOP ratio, the Blue Gene/Q provides an L1 cache prefetch engine for each core along with scalable atomic operations.

Connected to each node is a five dimensional (5-D) torus network with 10 serial links each capable of sending and receiving 2 GB/sec. The 5-D torus was chosen because it provides an overall best performance at a reasonable cost of implementation. An 11th link is dedicated to I/O which connects to the I/O node set. Unlike previous Blue Gene systems which have a dedicated I/O node for a set of compute nodes (e.g., pset), the Blue Gene/Q has a more flexible approach which pulls the I/O nodes outside of the primary compute fabric. In this configuration, up to eight I/O nodes, which are identical to the compute nodes from a hardware perspective, are grouped into a drawer. A single rack of Blue Gene/Q can drive up to 8 drawers (64 nodes) of I/O nodes. Each I/O node can service about 2 GB/sec of parallel file system data.

4.0 RESULTS AND DISCUSSION

4.1 Circuit Simulation

The circuit simulations were performed on two machines. The first is a 24-core symmetric multiprocessing (SMP) machine, operating at 2.667 GHz. The second is an IBM Blue Gene/L, with up to 1024 cores, each operating at 700 MHz. With the Blue Gene, ROSS is able to take advantage of the communication networks. In terms of computing power, 16 SMP cores is equivalent to 64 Blue Gene/L cores. This is due to the faster clock rates, a deep pipeline, and more functional units in the SMP's Intel X5650 processors. Throughout our testing, we focus on the event rate statistic of the simulation. This is principally due to the fact that the simulation is deterministic, resulting in consistent event counts for any sized simulation, irrespective of how many processors are used to execute it. Since each event represents a gate transition, the event rate metric reflects the speed of the overall simulation as well as the (high) volume of events we are able to compute. For all experiments, the base crossbar component consists of 211,001 individual gates.

4.1.1 Single Crossbar. The initial phase of testing consisted of a strong-scaling scenario: a single crossbar. Obviously, scale up occurred as the number of processors used increased. In Figures 4 and 5 a low volume of inputs was used, one wave of randomly generated inputs every 30 "clock cycles". We also tested a larger overall workload among all processors. Figures 6 and 7 shows a high volume of randomly generated input signals, one every two "clock cycles". These relatively small scale (211,001 gates) experiments generate approximately 1.4 billion events over 3000 simulated clock cycles.

In both of these experiments, it is surprising that the optimistic simulation did not result in better performance than the conservative simulation. This drop off in performance on the part of the optimistic simulation is due to the scant amount of work on each of the 1024 processors. Coupled with a high percentage of remote events, 62.56% in the high volume scenario, we have an overly optimistic execution.

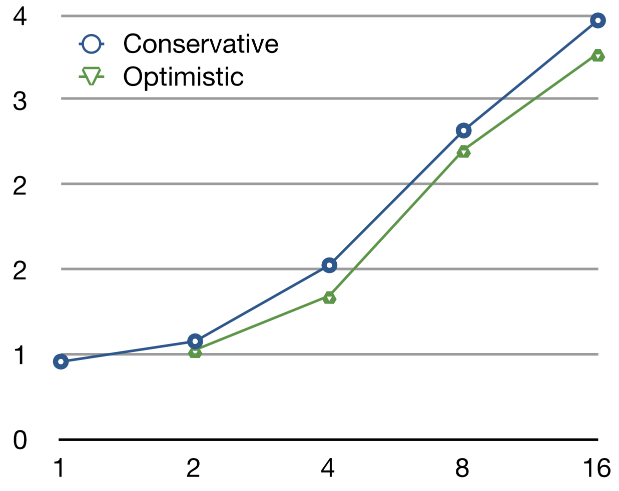

Figure 4 - Results from Low Volume, SMP Machine

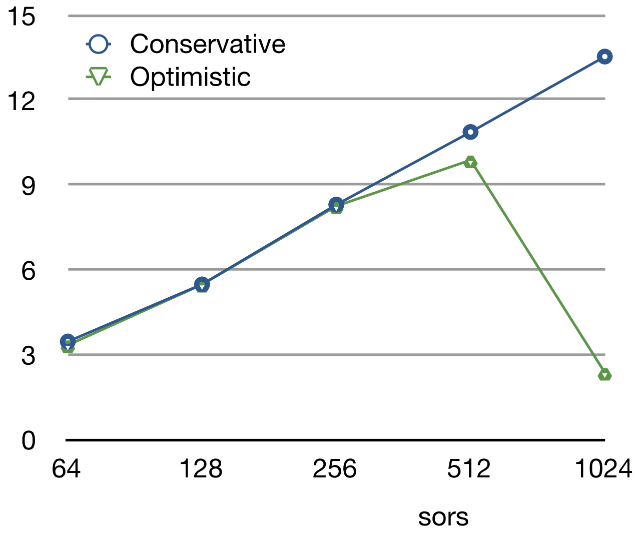

Figure 5 - Results from Low Volume, IBM Blue Gene/L

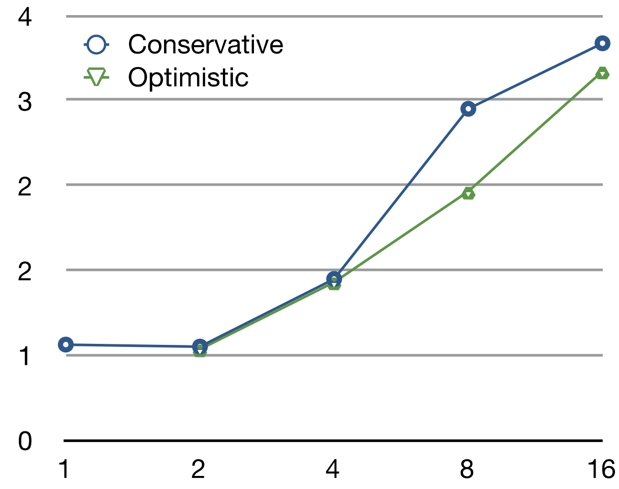

Figure 6 - Results from High Volume, SMP Machine

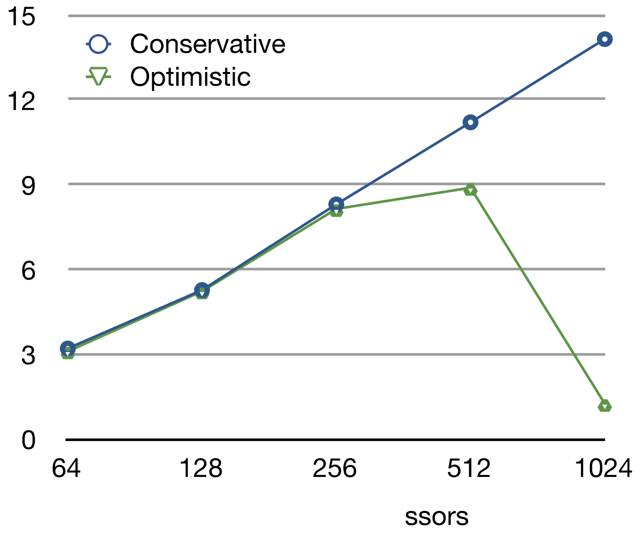

Figure 7 - Results from High Volume, IBM Blue Gene/L

4.1.2 Large Synthetic Circuits. The next experiment was a weak scaling study, as seen in Figures 8 and 9. This experiment clearly shows the power of the conservative simulation. The largest synthetic circuit, a 32x32 crossbar circuit (a single crossbar on each of the 1024 processors) consists of over 216 million gates. This conservative simulation contained 1.5 trillion events with an event rate of 116 million events per second. Gate-level models at this scale have, to the best of our knowledge, never been done before.

The result is an improvement in execution of at least 131- times. It should be noted that this is only a "conservative" estimate based upon the weak scaling experiments. Even if such a large simulation could be executed on a single core, we would expect to see super linear speedup when compared with a parallel system simulation because of cache memory effects.

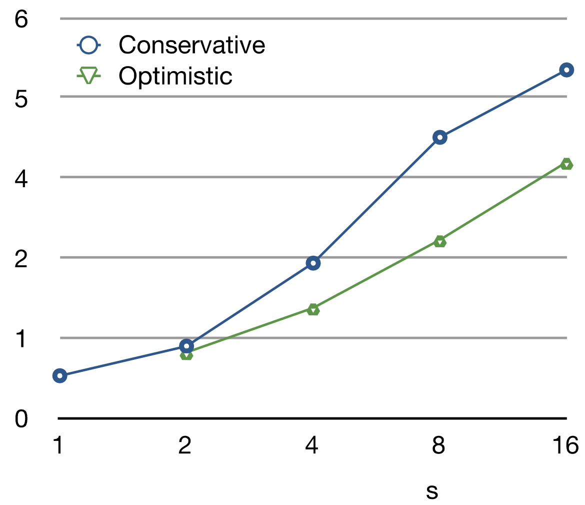

Figure 8 - Results from Weak Scaling, 24-Core SMP Machine

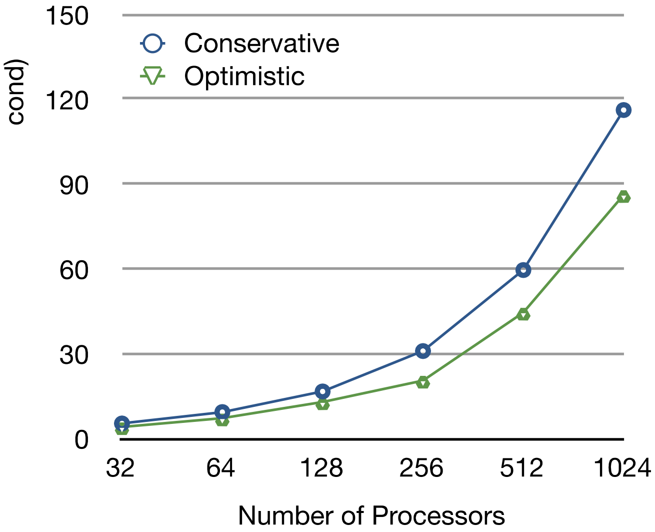

Figure 9 - Result from Weak Scaling, IBM Blue Gene/L

4.1.3 Optimistic Tuning. In an effort to fine-tune the simulation to improve the optimistic performance, we focused on reducing the forced GVT count. A forced GVT is usually an attempt by ROSS to reclaim some memory. Optimistic simulations usually take up more memory than their conservative counterparts due to the need to keep messages in case of rollbacks (and thus reverse computation). The following experiments were conducted on the 24-core SMP machine, using a 4x4 synthetic circuit on 16 cores (Figure 3).

The first experiment explores the effect of the size of the memory allocation (in terms of number of allocated events per processor), see Table 5.

Table 5 - Memory Allocation Effects on Forced GVT Count

Memory Allocated (Millions of Events)	Forced GVTs
0.6	1004
0.8	478
1.0	827
1.2	415

As is clear from this table, more allocated memory reduces the forced GVT count. However, memory is not a "free" resource, especially within supercomputing systems. Other variables must also be adjusted to achieve optimistic performance.

Within ROSS, the batch and GVT-interval parameters determine how many events are processed between successive global GVT computations. Batch specifies how many local events are processed before checking for network events. The GVT-interval defines how many event processing loops occur between GVT computations. The effect of these experiments on number of forced GVTs can be seen in Table 6. The high batch and GVT-interval product can be tolerated in this simulation due to the very small (less than 1%) remote event count.

We see here that forced GVTs are persistent and difficult to get rid of. Further attempts to improve optimistic performance will be part of our future research.

Table 6 - Effect of Batch and GVT-Interval Parameters on Forced GVT Count

Batch × GVT-Interval	Forced GVTs
2 × 1024	1312
4 × 1024	750
8 × 1024	577
16 × 1024	415
32 × 1024	393
32 × 2048	236
32 × 4096	195

4.2 Scaling Experiment

Our scaling experimental study includes two parallel systems. The first system is a two rack, 418 teraflop Blue Gene/Q system located at the CCNI. This system has 32,768 cores, 32TB of RAM and is configured with 64 I/O nodes (8 drawers) making it one of the most I/O rich configured Blue Genes for its size fielded today. The CCNI's Blue Gene/Q system is configured with driver level V1R2M0 Efix 13. This driver provides low level functionality to the Blue Gene/Q system especially related to the Parallel Active Message Interface (PAMI). The MPI implementation uses PAMI to efficiently transmit messages within the 5-D torus network. The ROSS compiler settings are: -qflag=i:i - qattr=full -O3 for all Blue Gene/Q runs.

The second system used for the scaling experiment is the Sequoia Blue Gene/Q system located at Lawrence Livermore National Laboratory (LLNL). This system is nominally comprised of 96 racks, 1,572,864 cores and nearly 1.6 PB of RAM yielding a peak performance of 20 petaflops. It is currently the number #2 ranked supercomputer in the world on the Top500 list (see: top500.org). For the commissioning period during which this study was conducted, Sequoia was joined with the Vulcan 24 rack Blue Gene/Q system, to form a 120 rack system with 1,966,080 cores and nearly 2 PB of RAM. While the compiler on Sequoia is the same as the CCNI system, our Sequoia experiments span two driver levels: V1R2M0 Efix 13 and V1R2M0 Efix 15. As we will demonstrate later, the Efix 15 driver enables ROSS to scale to 96 and 120 racks which was not possible with the Efix 13 driver. Additionally, for the Efix 15 runs, we had to add the following set of environment variable options in order to force PAMI and MPI to allocate the right amounts of internal memory storage: `BG_MAPCOMMONHEAP=1, BG_SHAREDMEMSIZE=128, PAMI_CLIENT_SHMEMSIZE=5M, PAMI_GLOBAL_SHMEMSIZE=70M, PAMI_SHMEM_NNODE_THRESHOLD=131072.`

We executed the PHOLD benchmark model, a derivative of the HOLD model, extended for parallel discrete event simulation and reverse computation across a number of parameters, including: `batch`, `GVT_interval`, MPI tasks per core (Blue Gene/Q only), and the percentage of remote communications. For review, the PHOLD model generates a synthetic workload by randomly either scheduling an event to “self” or some other randomly picked LP at an exponentially distributed time-stamp with mean of 1.0 into the future. An optional parameter, `lookahead`, can be added to the scheduled time to enable direct comparison between optimistic schedulers (which can handle events scheduled with no delay) and conservative schedulers (which need a lower bound time interval `lookahead` to schedule future events on remote LPs).

As part of our strong scaling performance study, all Sequoia runs were configured with 251,658,240 LPs, 16 events per LP, 8 KPs per MPI rank, 10% remote communication fraction, 0.1 `lookahead`, 8 `batch`, 512 `GVT_interval`, and 4 MPI tasks per core. We introduce a small amount of `lookahead`, 0.1, and set the mean of the exponential part of the future event time distribution to 0.90. The event itself has no substantial workload other than random number generation for selecting the destination LP and future time-stamp. At 120 racks and 7,864,320 MPI ranks (4 ranks per core) this yields a relatively small workload configuration of only 32 LPs and 512 live events total per MPI rank.

4.2.1 CCNI Blue Gene/Q Results. As previously indicated, these early tests are intended to mimic the same LP workloads that Sequoia has at 96 racks. For the CCNI Blue Gene/Q system all runs are configured with 5,242,880 LPs, 8 `batch` events, 512 `GVT_interval` and 0.10 `lookahead`.

In the first series of experiments, we investigate the impact of over-committing MPI tasks to each Blue Gene/Q core. Recall that the A2 processor supports up to four hardware thread contexts per core. Thus, it is possible to support 1, 2 or 4 MPI tasks or pthreads per core. Figure 10 reports the PHOLD event-rate performance as a function of the total number of MPI tasks for run sizes of 512, 1024 and 2048 nodes, which correspond to using a 1/2 rack, 1 full rack and 2 full racks. Unlike previous hyper-threaded architectures, such as those used by Intel, we observe nearly linear event-rate performance improvement as each node is con- figured to use 2 and 4 threads. We do observe less than linear scaling when going from 2 to 4 threads but it is significantly better than we have observed on our hyper-threaded platforms. The peak event rates are nearly 2 billion, 4 billion and 8 billion for 32,768, 65,536 and 131,072 MPI tasks respectively. In comparison to our previous peak PHOLD performance of just over 12 billion on 65,536 Blue Gene/P cores, we observe here that this particular model has 5x the LPs and events leading to greater overheads in the priority queue and event processing because of the larger memory footprint (e.g., more cache misses).

The next two figures report the PHOLD event-rate performance (Figure 11) and efficiency (Figure 12) as a function of node count across varying remote event percentages. While each remote event percentage curve has its own scaling slope, we do not observe any decrease in performance as the node count increases. This suggests that the 5-D torus network of the Blue Gene/Q enables the PHOLD model to continue to scale up to 131,072 MPI tasks. This is in spite of the large number of remote events; in one case 100% of the events are scheduled to remote LPs. We attribute this continued scaling to the greater workload on a per MPI task basis leading to increased availability of parallelism left within the PHOLD model to exploit. The peak event-rates at 131,072 MPI tasks for 10%, 25%, 50% and 100% remote event communications are 8 billion, 5 billion, 3 billion and nearly 2 billion, respectively. In terms of overall speedup, we observe that the slope of the 10% remote communication line is super-linear. We attribute this scalability to better cache performance as more cores are used for the same size problem.

The PHOLD model efficiency mirrors the event-rate performance as shown in Figure 12. Here, we observe that the overall efficiency ranges between 98% down to 93%. This suggests the Blue Gene/Q network is quickly moving messages between LPs so that rollback rates are reduced even at high remote event communication rates.

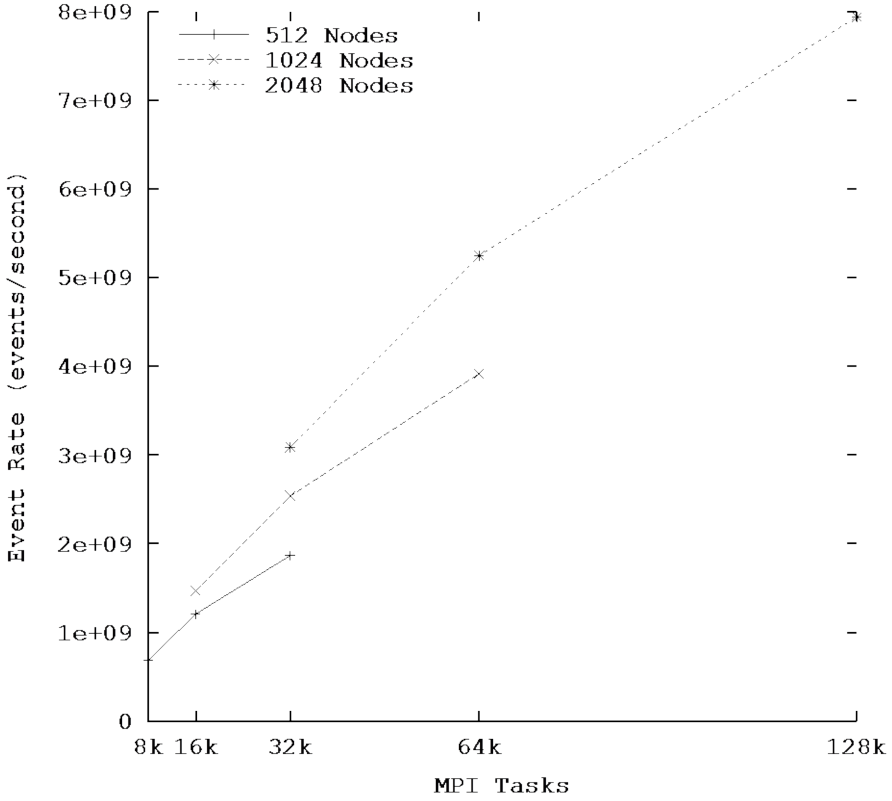

Figure 10 - CCNI: PHOLD Event-Rate Performance as a Function of MPI Tasks for Various Node Counts

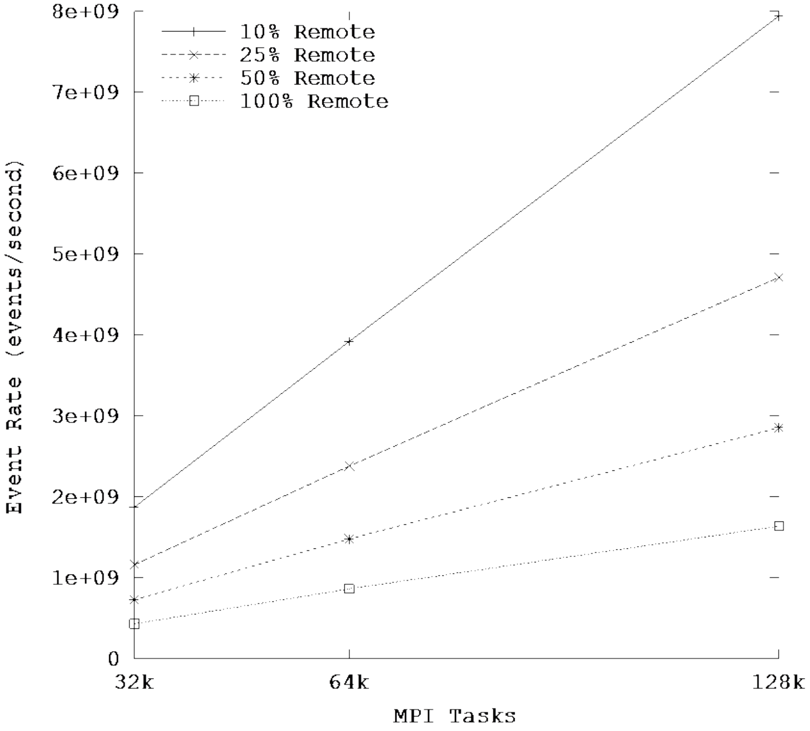

Figure 11 - CCNI: PHOLD Event-Rate Performance as a Function of MPI Tasks for Various Remote Percentages

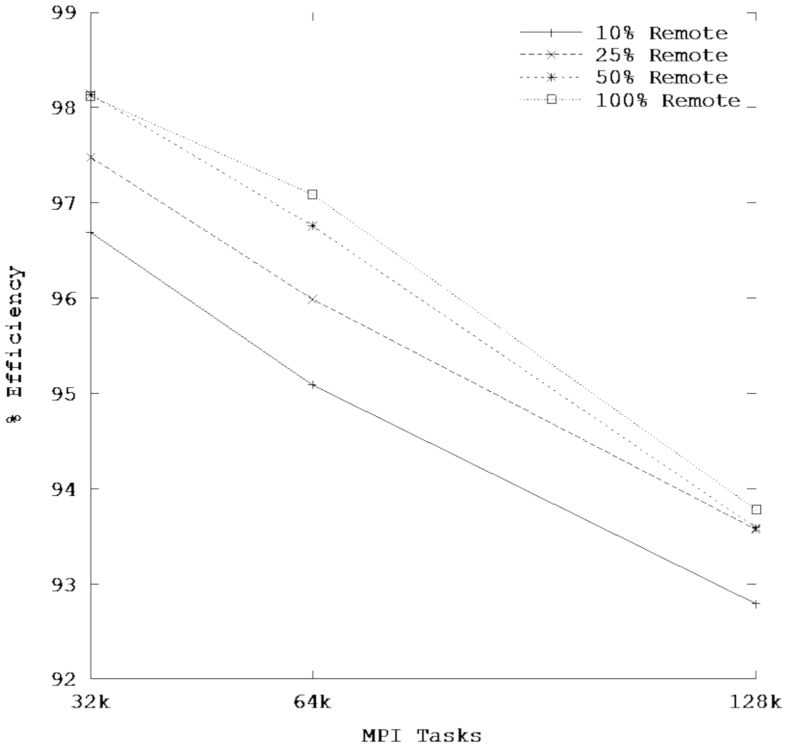

Figure 12 - CCNI: PHOLD Event Efficiency as a function of MPI Tasks for Various Remote Percentages

4.2.2 Sequoia Blue Gene/Q Results. As shown in Tables 7 and 8, we report the raw performance data collected from two distinct series of experiments on Sequoia. In Table 7, we report PHOLD performance using 1, 2, 4, 8, 48 racks for 3 runs each. The reason for the odd rack scaling (not strictly a power of 2) is attributed to the 5-D torus network which for a 96 rack system is a 16x16x16x12x2 network topology. In some configurations the torus becomes a mesh (e.g., no wrap around) which can increase message latency. However, these configurations above should always be a full torus.

Table 7 - SEQUOIA: Raw PHOLD Performance Data for 1, 2, 4, 8, and 48 Rack Runs

Cores	MPI	Total Events (x10^{13})	RB Events (x10^{11})	Net Events (x10^{13})	Efficiency %	Time (sec)	Event-Rate (x10^{10})	Warp
16,384	65,536	3.316	1.750	3.298	99.47	14964.185	0.220	0.34
16,384	65,536	3.316	1.750	3.298	99.47	15027.205	0.220	0.34
16,384	65,536	3.316	1.750	3.298	99.47	15027.660	0.219	0.34
32,768	131,072	3.325	2.678	3.298	99.19	6799.555	0.485	0.69
32,768	131,072	3.325	2.678	3.298	99.19	6822.710	0.483	0.68
32,768	131,072	3.325	2.678	3.298	99.19	6821.549	0.484	0.68
65,536	262,144	3.339	4.054	3.298	98.79	3128.452	1.05	1.02
65,536	262,144	3.339	4.054	3.298	98.79	3128.167	1.05	1.02
65,536	262,144	3.339	4.054	3.298	98.79	3127.627	1.05	1.02
131,072	524,288	3.358	5.988	3.298	98.22	1445122	2.28	1.36
131,072	524,288	3.358	5.988	3.298	98.22	1447.122	2.28	1.36
131,072	524,288	3.358	5.988	3.298	98.22	1447.125	2.28	1.36
786,432	3,145,728	3.455	15.62	3.298	95.48	202.581	16.3	2.21
786,432	3,145,728	3.455	15.63	3.298	95.48	200.728	16.4	2.22
786,432	3,145,728	3.455	15.62	3.298	95.48	200.750	16.4	2.22

Table 8 - SEQUOIA: Raw PHOLD Performance Data for 2, 8, 24, 48, 96, and 120 Rack Runs

Cores	MPI	Total Events (x10^{13})	RB Events (x10^{12})	Net Events (x10^{13})	Efficiency %	Time (sec)	Event-Rate (x10^{10})	Warp
32,768	131,072	3.325	0.2679	3.298	99.19	6377.8	0.517	0.71
32,768	131,072	3.325	0.2679	3.298	99.19	6378.3	0.517	0.71
65,536	262,144	3.339	0.4053	3.298	98.79	2912.4	1.13	1.05
65,536	262,144	3.339	0.4053	3.298	98.79	2912.4	1.13	1.05
131,072	524,288	3.358	0.5986	3.298	98.22	1323.8	2.49	1.40
131,072	524,288	3.358	0.5986	3.298	98.22	1323.0	2.49	1.40
393,216	1,572,864	3.407	1.084	3.298	96.82	379.7	8.69	1.94
393,216	1,572,864	3.407	1.084	3.298	96.82	379.7	8.69	1.94
786,432	3,145,728	3.454	1.554	3.298	95.50	174.7	18.9	2.28
786,432	3,145,728	3.454	1.554	3.298	95.50	174.7	18.9	2.28
1,572,864	6,291,456	3.521	2.222	3.298	93.69	82.6	39.9	2.60
1,572,864	6,291,456	3.521	2.222	3.298	93.69	82.6	39.9	2.60
1,966,080	7,864,320	3.547	2.490	3.298	92.98	65.4	50.4	2.70
1,966,080	7,864,320	3.548	2.490	3.298	92.98	65.5	50.4	2.70

In terms of performance, we observe very high efficiencies and record breaking event-rates. For example, at 1 rack (65,536 MPI tasks), the event-rate is just above 2.2 billion and requires just over 4 hours (4 hours, 10 minutes) to execute the 32 trillion event, 250 million LP PHOLD configuration. However, as we increase the core count, we observe PHOLD's event-rate not only increasing but doing so at a super-linear pace. At two racks, the event-rate has more than doubled to 4.85 billion and at 8 racks the event-rate is more than quadrupled to 22.8 billion. At 48 racks the event-rate is 164 billion. The speedup for 1 to 48 racks is 74x for this series of experiments. The overall simulator event efficiencies reflect the super-linear performance with a range of 93% to 99%. These experiments ran during the time window of 01/24/2013 to 02/05/2013. During that time, Sequoia's driver was Efix #13.

The second series of experiments shown in Table 8 were conducted during the time window of 03/08/2013 and 03/ 11/2013. At this time, Sequoia was running a much updated driver, Efix #15 that enable our PHOLD model to execute on 96 and 120 rack configurations. Here, we observe an unfathomable level of performance. First, Efix #15 version of the driver has improved our lower rack count performance. For example, the 2 rack performance improved from 4.85 to 5.17 billion events/second, which is nearly a 7% performance improvement. However, at the higher end, our 48 rack performance has improved by over 10 billion events per second (164.5 to 174.7). However, the *piéce de résistance* comes at 96 and 120 racks. Here, we observe event-rates of 399 billion and 504 billion respectively with an overall range speedup (2 to 120 racks) of 97x and both with an event efficiency of nearly 93%.

5.0 CONCLUSIONS

In this paper we have shown that gate-level simulation is possible for large circuits. A modestly sized (1024 node) supercomputer was used to demonstrate a 131-times performance increase when compared to a single core sequential simulation. In our most successful simulation, we were able to achieve 116 million gate-transaction events per second for a circuit of 216 million gates. We shall continue our research to improve our simulation speeds and to model more realistic circuits.

The lookahead of 0.4 time units (in relation to the timestamp increment of 0.5 time units) is in the medium to large range, based on the performance study by Carothers and Perumalla. From this study, we note that conservative synchronization outperforms optimistic synchronization for medium and large lookahead up to 8K cores. We observe the same phenomena here. Should the lookahead become smaller due to changes in the model or core count increase, we expect optimistic performance to improve and potentially overtake conservative performance. Future experimentation is required to confirm this hypothesis. A potential caveat for optimistic performance is high fan-out gate transitions that result in the scheduling of large volumes of events into the future. This places considerable pressure on memory and results in time consuming “forced” GVT computations. More experimentation is needed to determine the performance impact of these high fan-out event computations.

Through the scaling experiments we have demonstrated a new standard in Time Warp performance by efficiently executing the PHOLD benchmark model using ROSS on 1,966,080 Blue Gene/Q cores that yields a mind-blowing event-rate of over 504 billion. We emphasize that these performance results were obtained using a computational workload that generates a nearly worst-case, globally random communication patterns. From these performance results, we define a new long term performance metric called Warp Speed which grows linearly with an exponential increase in PHOLD event-rate performance. We are now at Warp Speed 2.7.

6.0 REFERENCES

Barnes, Jr., P., Jefferson, D., Carothers, C., and LaPre, J. “Warp Speed: Executing Time Warp on 1,966080 Cores,” *2013 ACM SIGSIM PADS*, (invited paper).

Gonsiorowski, E., Carothers, C., Tropper, C. “Modeling Large Scale Circuits Using Massively Parallel Discrete-Event Simulation,” *IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2012*). pp. 127-133. Arlington, Virginia, August 7–9th, 2012.

LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS

ANL	Argonne National Laboratory
CCNI	Rensselaer’s Computational Center for Nanotechnology Innovations
CCX	crossbar
GTECH	generic technology library
GVT	global virtual time
LLNL	Lawrence Livermore National Laboratory
LP	logical process
MPI	message passing interface
PAMI	parallel active message interface
PDES	parallel discrete-event simulation
ROSS	Rensselaer optimistic simulation system
RTL	register transfer language
SMP	symmetric multiprocessing